# Interface DS3231 RTC Module with STM32

Today in this tutorial we are going to interface DS3231 RTC module with STM32. The module works on I2C communication protocol, and therefore we need only 2 wires to interface it with the microcontroller.

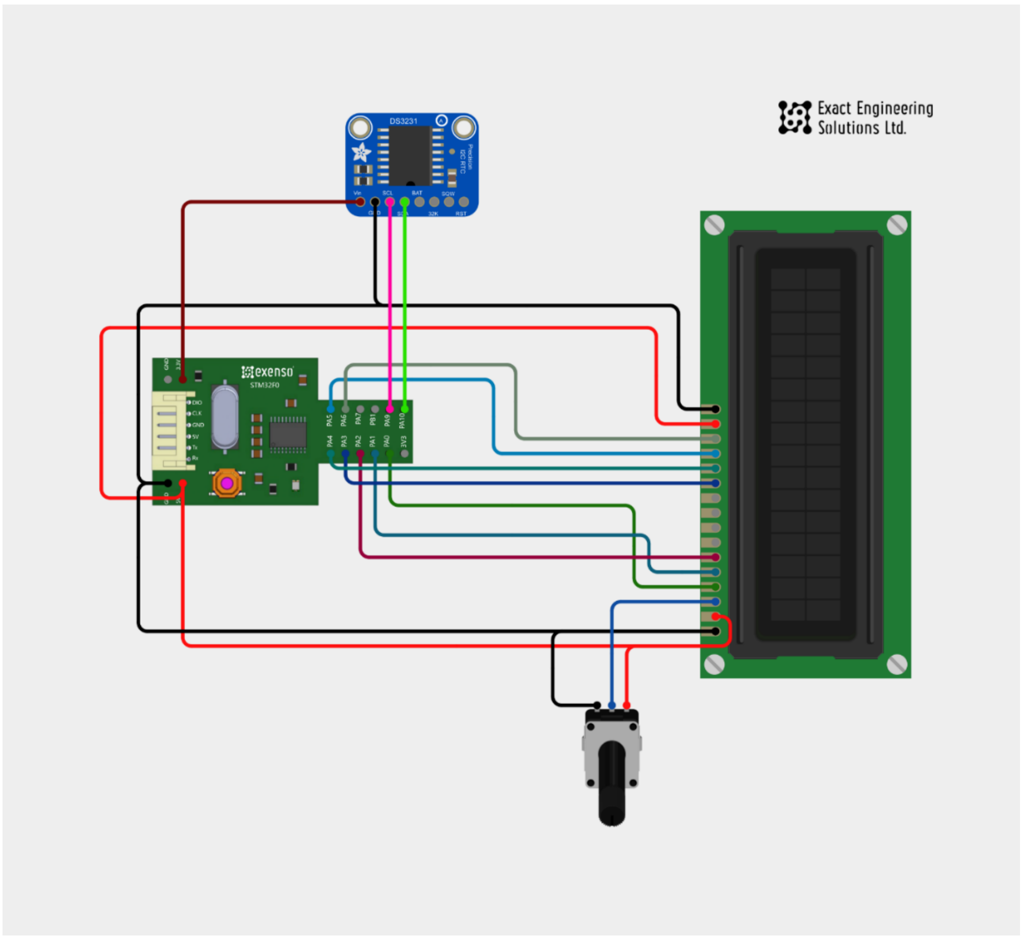
## Components Required

You will need the following components –

- 1 × Breadboard
- 1 × STM32F030F4P6
- 1× LCD 16x2
- 1× 10KΩ potentiometer
- 1x DS3231 RTC Modules
- Some Jumper wire

## Procedure

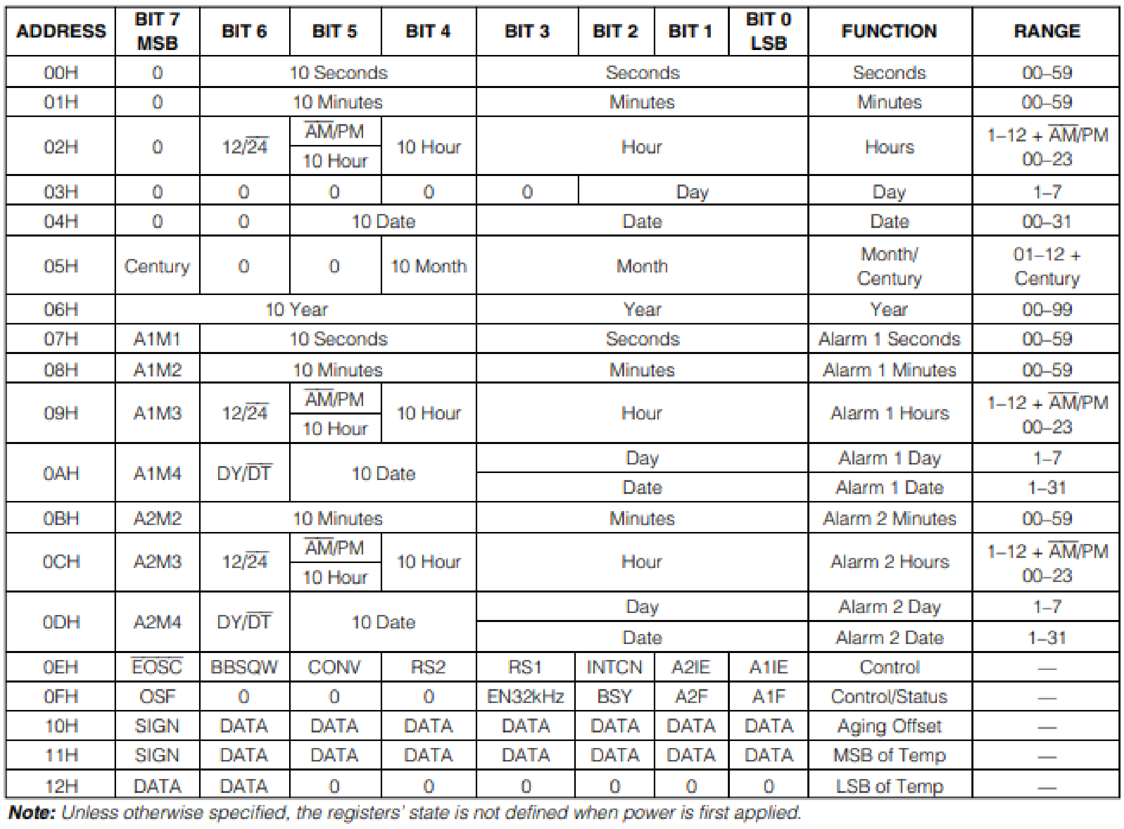
Follow the circuit diagram shown in the image given below.



There is nothing special about Interfacing DS3231 with any microcontroller. It’s basically a memory device, which we can write the data to and read the data from. Just like any other memory device, we have to do the following in order to perform the read/write operation

- Select the device by sending the device address on the I2C line
- Select the memory address that you want to access
- write/read the data to the address

The following is the picture from the datasheet of the device. It shows the registers available in the DS3231 for writing and reading data

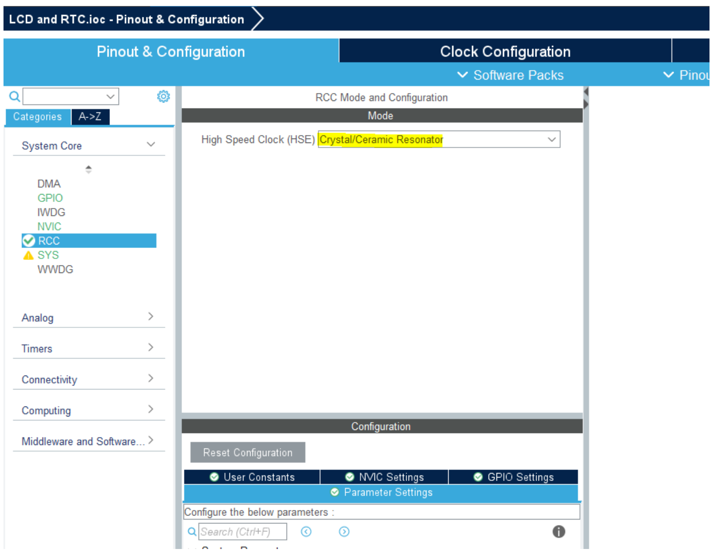


| ADDRESS | BIT 7 MSB | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 LSB | FUNCTION | RANGE |
|---|---|---|---|---|---|---|---|---|---|---|
| 00H | 0 | 10 Seconds | | | Seconds | | | | Seconds | 00–59 |
| 01H | 0 | 10 Minutes | | | Minutes | | | | Minutes | 00–59 |
| 02H | 0 | 12/$\overline{24}$ | $\overline{\text{AM}}$/PM 10 Hour | 10 Hour | Hour | | | | Hours | 1–12 + $\overline{\text{AM}}$/PM 00–23 |
| 03H | 0 | 0 | 0 | 0 | 0 | Day | | | Day | 1–7 |
| 04H | 0 | 0 | 10 Date | | Date | | | | Date | 00–31 |
| 05H | Century | 0 | 0 | 10 Month | Month | | | | Month/ Century | 01–12 + Century |
| 06H | 10 Year | | | | Year | | | | Year | 00–99 |
| 07H | A1M1 | 10 Seconds | | | Seconds | | | | Alarm 1 Seconds | 00–59 |
| 08H | A1M2 | 10 Minutes | | | Minutes | | | | Alarm 1 Minutes | 00–59 |
| 09H | A1M3 | 12/$\overline{24}$ | $\overline{\text{AM}}$/PM 10 Hour | 10 Hour | Hour | | | | Alarm 1 Hours | 1–12 + $\overline{\text{AM}}$/PM 00–23 |
| 0AH | A1M4 | DY/$\overline{\text{DT}}$ | 10 Date | | Day | | | | Alarm 1 Day | 1–7 |
| | | | | | Date | | | | Alarm 1 Date | 1–31 |
| 0BH | A2M2 | 10 Minutes | | | Minutes | | | | Alarm 2 Minutes | 00–59 |
| 0CH | A2M3 | 12/$\overline{24}$ | $\overline{\text{AM}}$/PM 10 Hour | 10 Hour | Hour | | | | Alarm 2 Hours | 1–12 + $\overline{\text{AM}}$/PM 00–23 |
| 0DH | A2M4 | DY/$\overline{\text{DT}}$ | 10 Date | | Day | | | | Alarm 2 Day | 1–7 |
| | | | | | Date | | | | Alarm 2 Date | 1–31 |
| 0EH | $\overline{\text{EOSC}}$ | BBSQW | CONV | RS2 | RS1 | INTCN | A2IE | A1IE | Control | — |
| 0FH | OSF | 0 | 0 | 0 | EN32kHz | BSY | A2F | A1F | Control/Status | — |
| 10H | SIGN | DATA | DATA | DATA | DATA | DATA | DATA | DATA | Aging Offset | — |
| 11H | SIGN | DATA | DATA | DATA | DATA | DATA | DATA | DATA | MSB of Temp | — |
| 12H | DATA | DATA | 0 | 0 | 0 | 0 | 0 | 0 | LSB of Temp | — |

***Note:*** *Unless otherwise specified, the registers' state is not defined when power is first applied.*

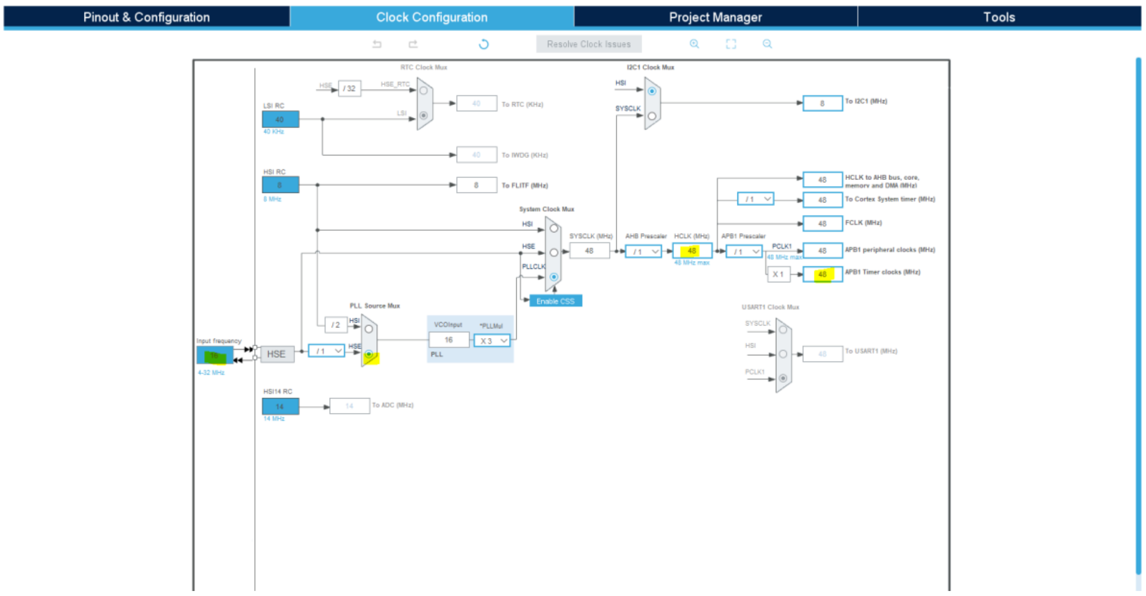
As I mentioned, in this tutorial we are only going to interface the RTC part. therefore we are interested in the registers ranging from address 00h to 06h for the clock and date.

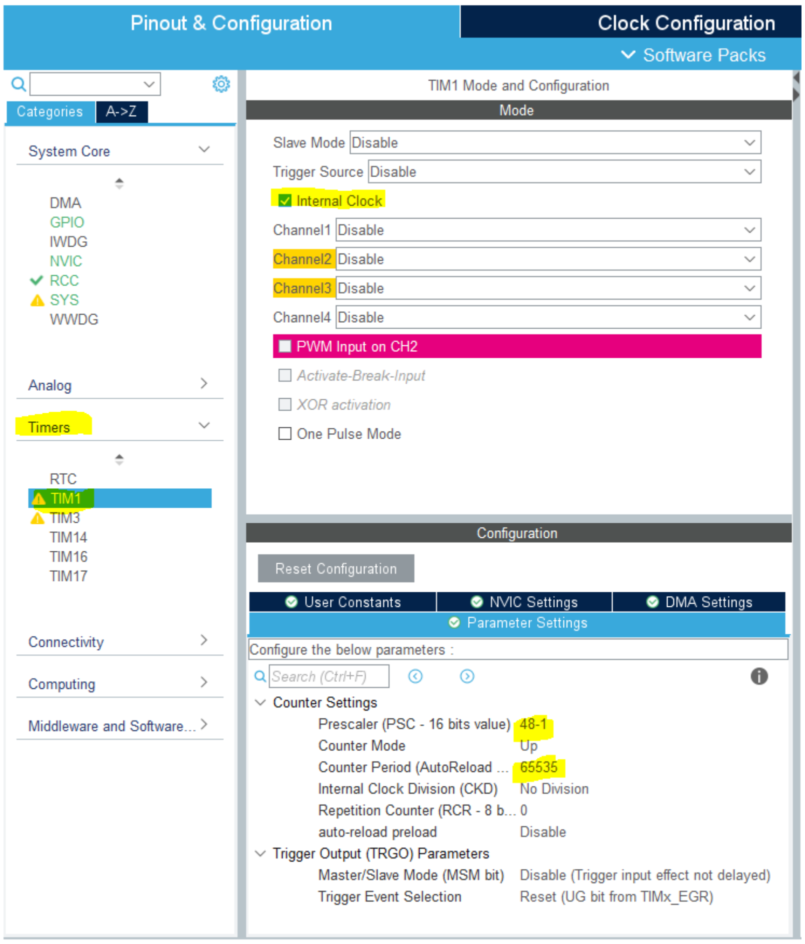
## STM32F0 Pin Configuration:



High Speed Clock (HSE) Crystal/Ceramic Resonator

Configure the below parameters :





TIM1 Mode and Configuration

Mode

Slave Mode Disable

Trigger Source Disable

☑ Internal Clock

Channel1 Disable

Channel2 Disable

Channel3 Disable

Channel4 Disable

☐ PWM Input on CH2

☐ *Activate-Break-Input*

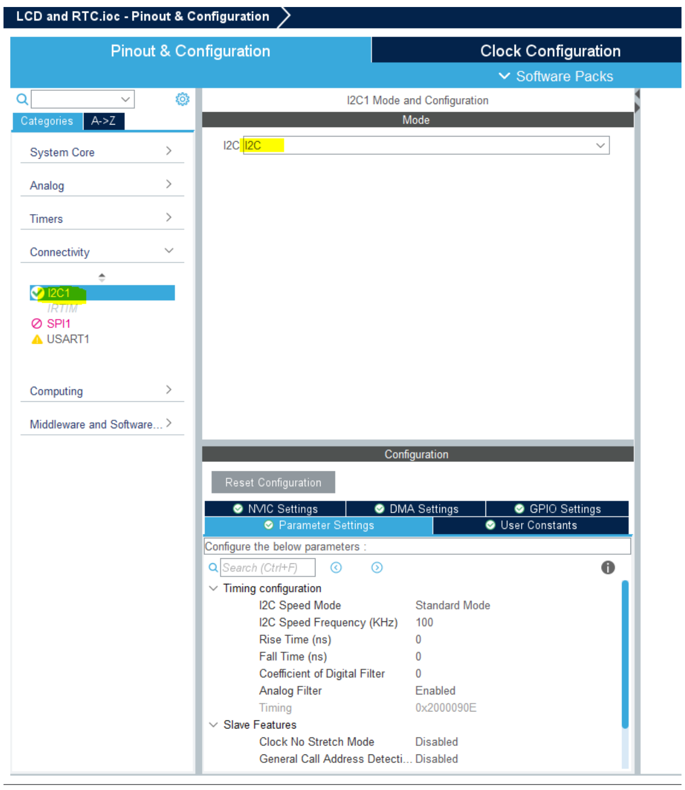
☐ *XOR activation*

☐ One Pulse Mode

Configuration

Configure the below parameters :

```
Counter Settings
    Prescaler (PSC - 16 bits value)     48-1
    Counter Mode                        Up
    Counter Period (AutoReload ...      65535
    Internal Clock Division (CKD)       No Division
    Repetition Counter (RCR - 8 b...    0
    auto-reload preload                 Disable
Trigger Output (TRGO) Parameters
    Master/Slave Mode (MSM bit)         Disable (Trigger input effect not delayed)
    Trigger Event Selection             Reset (UG bit from TIMx_EGR)
```


LCD and RTC.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Software Packs

Categories | A->Z

System Core
Analog
Timers
Connectivity
- I2C1
- IRTIM
- SPI1
- USART1

Computing
Middleware and Software...

I2C1 Mode and Configuration

Mode

I2C: I2C

Configuration

Reset Configuration

NVIC Settings | DMA Settings | GPIO Settings | Parameter Settings | User Constants

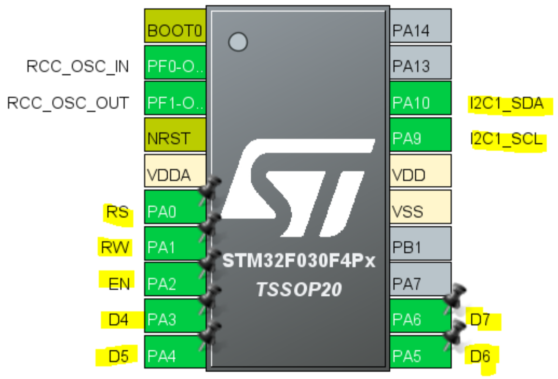
Configure the below parameters :

Timing configuration

| | |
|---|---|
| I2C Speed Mode | Standard Mode |
| I2C Speed Frequency (KHz) | 100 |
| Rise Time (ns) | 0 |
| Fall Time (ns) | 0 |
| Coefficient of Digital Filter | 0 |
| Analog Filter | Enabled |
| Timing | 0x2000090E |

Slave Features

| | |
|---|---|
| Clock No Stretch Mode | Disabled |
| General Call Address Detecti... | Disabled |



Pin PA0 To PA6 set as Output

## Code

```c
#include "main.h"
#include "LCD1602.h"
#include "DS321RTC.h"

I2C_HandleTypeDef hi2c1;

TIM_HandleTypeDef htim1;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_TIM1_Init(void);


void Print(int32_t num) {
  char string[4];
  string[0] = ( (num/10) % 10) + 48;
  string[1] = ( num % 10) + 48;
  string[2] = 0;
  LCD_String(string); // Send the formatted string to your LCD display function

  LCD_String(" ");
}

int main(void)
{
  HAL_Init();

  SystemClock_Config();

  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_TIM1_Init();

  HAL_TIM_Base_Start(&htim1);  // Timer On and init this line
  lcd_init();                 //LCD init
  Set_Time(00,30,2,1,4,5,24); // Set_Time(sec,min,hour,dow,,dom,month,year);

  lcd_xy(0, 0);
  LCD_String("Time :");
  lcd_xy(1, 0);
  LCD_String("Date :");

  while (1)
  {
        Get_Time ();
        lcd_xy(0, 7);
        Print(time.hour);
        lcd_xy(0, 9);
        LCD_String(":");
        Print(time.minutes);
        lcd_xy(0, 12);
        LCD_String(":");
        Print(time.seconds);
```

```
         lcd_xy(1, 7);
        Print(time.dayofmonth);
        lcd_xy(1, 9);
        LCD_String("-");
        Print(time.month);
        lcd_xy(1, 12);
        LCD_String("-");
        Print(time.year);
  }

}
```

We have to set the time once in the main function, and then we can always read the time in the while loop.

One more very important thing. After setting the time, you must flash the code again with the Set_Time function as commented out as shown below

```
  //Set_Time(00,30,2,1,4,5,24);
```

So, just comment out the function and flash again. This is to ensure that, if the microcontroller resets, the RTC won't start from this time again.

**Output:**